J 5

```
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
CCC              DDD         DDD  UUU         UUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
   CCCCCCCCCCCC  DDDDDDDDDDDD     UUUUUUUUUUUUUUU
```

```
**FILE**ID**MAIN
```

```
MM      MM    AAAAAA      IIIIII    NN      NN
MM      MM    AAAAAA      IIIIII    NN      NN
MMMM  MMMM  AA      AA      II      NN      NN
MMMM  MMMM  AA      AA      II      NN      NN
MM  MM  MM  AA      AA      II      NNNN    NN
MM  MM  MM  AA      AA      II      NNNN    NN
MM      MM  AA      AA      II      NN  NN  NN
MM      MM  AA      AA      II      NN  NN  NN
MM      MM  AAAAAAAAAA      II      NN    NNNN
MM      MM  AAAAAAAAAA      II      NN    NNNN
MM      MM  AA      AA      II      NN      NN      ....
MM      MM  AA      AA      II      NN      NN      ....
MM      MM  AA      AA    IIIIII    NN      NN      ....
MM      MM  AA      AA    IIIIII    NN      NN      ....

LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II      SS
LL                II        SSSSSS
LL                II        SSSSSS
LL                II              SS
LL                II              SS
LL                II              SS
LL                II              SS
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
0 MODULE main               (IDENT='V04-000',
0                            MAIN=CDU$MAIN,
0                            ADDRESSING_MODE(EXTERNAL=GENERAL))
1 = BEGIN
1
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 ! Facility:     Command Definition Utility, Main Module
1 !
1 ! Abstract:     This module contains the main routines for the Command
1 !               Definition Utility, formerly known as the Command Language
1 !               Editor.  The CDU is responsible for maintaining CLI Tables,
1 !               which are images or object files containing the internal
1 !               representation of DCL or MCR commands.  The primary
1 !               component of the CDU is a compiler which reads Command
1 !               Language Definition (CLD) files and compiles them into the
1 !               internal format.  Other features allow the deletion and
1 !               extraction of information from DCL Tables, plus other
1 !               goodies.
1 !
1 !               Special thanks goes to Tim Halvorsen, who wrote the
1 !               original CDU.  It has been rewritten to make it a bit more
1 !               flexible and easy to maintain, particularly in light of all
1 !               the enhancements in VMS V4.
1 !
1 ! Environment:  Native, User mode.  The following privileges are required:
1 !
1 !                       CMEXEC          For fooling with P1 space.
1 !
1 ! Author:       Paul C. Anagnostopoulos
1 ! Creation:     18 January 1983
1 !
1 ! Modifications:
1 !--
1
```

```
;    58        0058  1
;    59        0059  1 library 'sys$library:lib';
;    60        0060  1 require 'cdureq';
;    62          0474  1 !       T A B L E   O F   C O N T E N T S
;    63          0475  1 !       ---------   ---   ---------------
;    64          0476  1
;    65          0477  1 forward routine
;    66          0478  1         cdu$main,
;    67          0479  1         cdu$delete_mode: novalue,
;    68          0480  1         cdu$object_mode: novalue,
;    69          0481  1         cdu$replace_mode: novalue,
;    70          0482  1         cdu$symbols_mode: novalue,
;    71          0483  1         cdu$report_rms_error: novalue;
;    72          0484  1
;    73          0485  1
;    74          0486  1 !       E X T E R N A L   R E F E R E N C E S
;    75          0487  1 !       ---------------   -------------------
;    76          0488  1
;    77          0489  1 external routine
;    78          0490  1         cdu$cld,
;    79          0491  1         cdu$close_symbol_table_file,
;    80          0492  1         cdu$delete_verb_name,
;    81          0493  1         cdu$free_all_nodes,
;    82          0494  1         cdu$generate_table_blocks,
;    83          0495  1         cdu$open_next_cld,
;    84          0496  1         cdu$prepare_input_table,
;    85          0497  1         cdu$prepare_listing_file,
;    86          0498  1         cdu$prepare_new_table,
;    87          0499  1         cdu$prepare_object_file,
;    88          0500  1         cdu$report_listing_trailer,
;    89          0501  1         cdu$write_object_file,
;    90          0502  1         cdu$write_output_table,
;    91          0503  1         cdu$write_symbol_table_file,
;    92          0504  1         cli$get_value,
;    93          0505  1         cli$present,
;    94          0506  1         str$trim;
;    95          0507  1
;    96          0508  1 external
;    97          0509  1         cdu$gl_cld_errors: long;
```

```
1 !       G L O B A L   D A T A
1 !       -----------   -------
1
1 ! The following item specifies the facility string to be used in object files
1 ! or any other files we create.
1
1 global bind
1         cdu$facility_string = dtext('VAX/VMS Command Definition Utility (V4-001)'): descriptor;
```

```
;  108       0518  1 !++
;  109       0519  1 ! Description:  This is the main routine of the Command Definition Utility.
;  110       0520  1 !               It is responsible for determining which operating mode the
;  111       0521  1 !               user has requested and invoking a routine for that mode.
;  112       0522  1 !
;  113       0523  1 ! Parameters:   None.
;  114       0524  1 !
;  115       0525  1 ! Returns:      Most severe status encountered during execution.
;  116       0526  1 !
;  117       0527  1 ! Notes:
;  118       0528  1 !--
;  119       0529  1
;  120       0530  1 GLOBAL ROUTINE cdu$main
;  121       0531  2 = BEGIN
;  122       0532  2
;  123       0533  2 own
;  124       0534  2         worst_status: long initial(msg(cdu$_success));
;  126        0535  2 ! The following routine is the global condition handler.  Its purpose is to
;  127        0536  2 ! save the worst status that is signalled during the execution of the CDU.
;  128        0537  2 ! It is this status that is returned to DCL.
;  129        0538  2
;  130        0539  2 ROUTINE condition_handler(signal_vector: ref vector[,long])
;  131        0540  3 = BEGIN
;  132        0541  3
;  133        0542  3 bind
;  134        0543  3         status = signal_vector[1]: long;
;  135        0544  3 own
;  136        0545  3         severity_map: vector[8,byte] initial(byte(2,0,3,1,4,4,4,4));
;  137        0546  3
;  138        0547  3 if .severity_map[.status<0,3,0>] gtru .severity_map[.worst_status<0,3,0>] then
;  139        0548  3         worst_status = .status;
;  140        0549  3
;  141        0550  3 return false;
;  142        0551  3
;  143        0552  2 END;


                                                                        .TITLE  MAIN
                                                                        .IDENT  \V04-000\

                                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

64 6E 61 6D 6D 6F 43 20 53 4D 56 2F 58 41 56  00000 P.AAB:  .ASCII  \VAX/VMS Command Definition Utility (V4-0\  ;
69 74 55 20 6E 6F 69 74 69 6E 69 66 65 44 20  0000F                                                              ;
               30 2D 34 56 28 20 79 74 69 6C  0001E                                                              ;
                              00 29 31 30  00028         .ASCII  \01)\<0>                                          ;
                               010E002B  0002C P.AAA:  .LONG   17694763                                          ;
                               00000000' 00030         .ADDRESS P.AAB                                            ;

                                                                        .PSECT  $OWN$,NOEXE,2

                               00000000G 00000 WORST_STATUS:
                                                        .LONG   CDU$_SUCCESS                                      ;
            04 04 04 04 01 03 00 02  00004 SEVERITY_MAP:
                                                        .BYTE   2, 0, 3, 1, 4, 4, 4, 4                            ;

                                               CDU$FACILITY_STRING==
                                                                 P.AAA
                                                        .EXTRN  CDU$CLD, CDU$CLOSE_SYMBOL_TABLE_FILE
                                                        .EXTRN  CDU$DELETE_VERB_NAME
                                                        .EXTRN  CDU$FREE_ALL_NODES
                                                        .EXTRN  CDU$GENERATE_TABLE_BLOCKS
                                                        .EXTRN  CDU$OPEN_NEXT_CLD
                                                        .EXTRN  CDU$PREPARE_INPUT_TABLE
                                                        .EXTRN  CDU$PREPARE_LISTING_FILE
                                                        .EXTRN  CDU$PREPARE_NEW_TABLE
                                                        .EXTRN  CDU$PREPARE_OBJECT_FILE
                                                        .EXTRN  CDU$REPORT_LISTING_TRAILER
                                                        .EXTRN  CDU$WRITE_OBJECT_FILE
                                                        .EXTRN  CDU$WRITE_OUTPUT_TABLE
                                                        .EXTRN  CDU$WRITE_SYMBOL_TABLE_FILE
                                                        .EXTRN  CLI$GET_VALUE, CLI$PRESENT
                                                        .EXTRN  STR$TRIM, CDU$GL_CLD_ERRORS
                                                        .EXTRN  CDU$_SUCCESS
```

```
                                                                    .PSECT  $CODE$,NOWRT,2

                                             000C 00000 CONDITION_HANDLER:
                                                                    .WORD   Save R2,R3                                  ; 0539
                              53     0000'   CF  9E 00002           MOVAB   WORST_STATUS, R3                            ;
                  52      04  AC             04  C1 00007           ADDL3   #4, SIGNAL_VECTOR, R2                       ; 0543
     51           62          03             00  EF 0000C           EXTZV   #0, #3, (R2), R1                            ; 0547
     50           63          03             00  EF 00011           EXTZV   #0, #3, WORST_STATUS, R0                    ;
                          04 A340      04 A341   91 00016           CMPB    SEVERITY_MAP[R1], SEVERITY_MAP[R0]          ;
                                             03  1B 0001D           BLEQU   1$                                          ;
                              63             62  D0 0001F           MOVL    (R2), WORST_STATUS                          ; 0548
                                             50  D4 00022 1$:       CLRL    R0                                          ; 0550
                                                 04 00024           RET                                                 ; 0552

; Routine Size:  37 bytes,    Routine Base:  $CODE$ + 0000
;   145        0553  2 ! Main routine.
;   146        0554  2 ! Establish a global condition handler to save the worst status that is
;   147        0555  2 ! signalled.
;   148        0556  2
;   149        0557  2 enable
;   150        0558  2         condition_handler;
;   151        0559  2
;   152        0560  2 ! Determine which operating mode the user wants.  This is specified by a
;   153        0561  2 ! major qualifier on the SET COMMAND command, of which /REPLACE is the default.
;   154        0562  2
;   155        0563  2 if cli$present(dtext('DELETE')) then
;   156        0564  2         cdu$delete_mode()
;   157        0565  2 else if cli$present(dtext('OBJECT')) then
;   158        0566  2         cdu$object_mode()
;   159        0567  2 else if cli$present(dtext('SYMBOLS')) then
;   160        0568  2         cdu$symbols_mode()
;   161        0569  2 else
;   162        0570  2         cdu$replace_mode();
;   163        0571  2
;   164        0572  2 ! Return the worst status that was signalled, with the inhibit flag set.
;   165        0573  2
;   166        0574  2 return .worst_status + sts$m_inhib_msg;
;   167        0575  2
;   168        0576  1 END;


                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2

          00 00 45 54 45 4C 45 44  00034 P.AAD:         .ASCII  \DELETE\<0><0>                              ;
                         010E0006  0003C P.AAC:         .LONG   17694726                                    ;
                        00000000'  00040                .ADDRESS P.AAD                                      ;
          00 00 54 43 45 4A 42 4F  00044 P.AAF:         .ASCII  \OBJECT\<0><0>                              ;
                         010E0006  0004C P.AAE:         .LONG   17694726                                    ;
                        00000000'  00050                .ADDRESS P.AAF                                      ;
          00 53 4C 4F 42 4D 59 53  00054 P.AAH:         .ASCII  \SYMBOLS\<0>                                ;
                         010E0007  0005C P.AAG:         .LONG   17694727                                    ;
                        00000000'  00060                .ADDRESS P.AAH                                      ;


                                                        .PSECT  $CODE$,NOWRT,2

                             0004  00000                .ENTRY  CDU$MAIN, Save R2                           ; 0530
             52 00000000G  00  9E  00002                MOVAB   CLI$PRESENT, R2                             ;
             6D       0044 CF  DE  00009                MOVAL   5$, (FP)                                    ; 0531
                     0000' CF  9F  0000E                PUSHAB  P.AAC                                       ; 0563
             62            01  FB  00012                CALLS   #1, CLI$PRESENT                             ;
             07            50  E9  00015                BLBC    R0, 1$                                      ;
       0000V CF            00  FB  00018                CALLS   #0, CDU$DELETE_MODE                         ; 0564
                           27  11  0001D                BRB     4$                                          ;
                     0000' CF  9F  0001F 1$:            PUSHAB  P.AAE                                       ; 0565
             62            01  FB  00023                CALLS   #1, CLI$PRESENT                             ;
             07            50  E9  00026                BLBC    R0, 2$                                      ;
       0000V CF            00  FB  00029                CALLS   #0, CDU$OBJECT_MODE                         ; 0566
                           16  11  0002E                BRB     4$                                          ;
                     0000' CF  9F  00030 2$:            PUSHAB  P.AAG                                       ; 0567
```

```
                   62            01  FB 00034           CALLS   #1, CLI$PRESENT                 ;
                   07            50  E9 00037           BLBC    R0, 3$                          ;
            0000V  CF            00  FB 0003A           CALLS   #0, CDU$SYMBOLS_MODE            ; 0568
                   05            11     0003F           BRB     4$                              ;
            0000V  CF            00  FB 00041 3$:       CALLS   #0, CDU$REPLACE_MODE            ; 0570
      50    0000'  CF 10000000   8F  C1 00046 4$:       ADDL3   #268435456, WORST_STATUS, R0    ; 0574
                                 04     00050           RET                                     ; 0576
                               0000     00051 5$:       .WORD   Save nothing                    ; 0531
                                 7E  D4 00053           CLRL    -(SP)                           ;
                                 5E  DD 00055           PUSHL   SP                              ;
                   7E        04  AC  7D 00057           MOVQ    4(AP), -(SP)                    ;
            FF7B   CF            03  FB 0005B           CALLS   #3, CONDITION_HANDLER           ;
                                 04     00060           RET                                     ;

; Routine Size:  97 bytes,    Routine Base:  $CODE$ + 0025
```

```
!++
! Description:  This routine handles /DELETE mode, in which the user wants
!               to remove one or more verb names from the CLI table.  We
!               retrieve the list of verb names and delete them from the
!               table, reporting any errors.
!
! Parameters:   None.
!
! Returns:      Nothing.
!
! Notes:
!--

GLOBAL ROUTINE cdu$delete_mode          : novalue
= BEGIN

local
        status: long;


! Call a routine to prepare the input CLI table for modification.

cdu$prepare_input_table();

! Loop through the list of verb names to be deleted.

loop (

        ! We need a buffer with descriptor to get a verb name.

        with_dbuffer(verb_name,32,

                ! Get the next verb name in the list.  Quit if there aren't
                ! any more.

                status = cli$get_value(dtext('DELETE'),verb_name);
                if not .status then exitloop;
                str$trim(verb_name,verb_name,verb_name);

                ! Call a routine to delete the verb name from the table.

                status = cdu$delete_verb_name(verb_name);
                check(.status, msg(cdu$_nosuchverb),1,verb_name);
        );
);

! Write out the modified CLI table.

cdu$write_output_table();

return;

END;
```

```
                                                        .PSECT  $PLIT$,NOWRT,NOEXE,2
```

```
00  00  45  54  45  4C  45  44  00064 P.AAJ:   .ASCII   \DELETE\<0><0>                          ;
                     010E0006   0006C P.AAI:   .LONG    17694726                                ;
                     00000000'  00070          .ADDRESS P.AAJ                                   ;

                                               .EXTRN   CDU$_NOSUCHVERB

                                               .PSECT   $CODE$,NOWRT,2

                         0004   00000          .ENTRY   CDU$DELETE_MODE, Save R2                ; 0590
             5E            28 C2 00002         SUBL2    #40, SP                                 ;
   00000000G 00            00 FB 00005         CALLS    #0, CDU$PREPARE_INPUT_TABLE             ; 0599
             6E            20 D0 0000C 1$:     MOVL     #32, VERB_NAME                          ; 0620
       04    AE       08   AE 9E 0000F         MOVAB    VERB_NAME+8, VERB_NAME+4                ;
                           5E DD 00014         PUSHL    SP                                      ;
                    0000'  CF 9F 00016         PUSHAB   P.AAI                                   ;
   00000000G 00            02 FB 0001A         CALLS    #2, CLI$GET_VALUE                       ;
             52            50 D0 00021         MOVL     R0, STATUS                              ;
             31            52 E9 00024         BLBC     STATUS, 2$                              ;
                           5E DD 00027         PUSHL    SP                                      ;
                      04   AE 9F 00029         PUSHAB   VERB_NAME                               ;
                      08   AE 9F 0002C         PUSHAB   VERB_NAME                               ;
   00000000G 00            03 FB 0002F         CALLS    #3, STR$TRIM                            ;
                           5E DD 00036         PUSHL    SP                                      ;
   00000000G 00            01 FB 00038         CALLS    #1, CDU$DELETE_VERB_NAME                ;
             52            50 D0 0003F         MOVL     R0, STATUS                              ;
             C7            52 E8 00042         BLBS     STATUS, 1$                              ;
                           5E DD 00045         PUSHL    SP                                      ;
                           01 DD 00047         PUSHL    #1                                      ;
                00000000G  8F DD 00049         PUSHL    #CDU$_NOSUCHVERB                        ;
   00000000G 00            03 FB 0004F         CALLS    #3, LIB$SIGNAL                          ;
                           B4 11 00056         BRB      1$                                      ; 0599
   00000000G 00            00 FB 00058 2$:     CALLS    #0, CDU$WRITE_OUTPUT_TABLE              ; 0625
                              04 0005F         RET                                              ; 0629

; Routine Size:  96 bytes,    Routine Base:  $CODE$ + 0086
```

```
!++
! Description:  This routine handles /OBJECT mode, in which the user wants
!               to compile an object file representing one CLD file.  The
!               CLD file is compiled and the resulting table blocks are
!               written into an object file.
!
! Parameters:   None.
!
! Returns:      Nothing.
!
! Notes:
!--

GLOBAL ROUTINE cdu$object_mode          : novalue
= BEGIN

local
        cld_fab: pointer,
        first_cld: boolean initial(true);


! Call a routine to set up a new, empty CLI table.  Commands defined in the
! CLD file will be added to this table.

cdu$prepare_new_table();

! Open the CLD file.  If there isn't one, forget it.

cld_fab = cdu$open_next_cld();
if .cld_fab eqla 0 then
        return;

! Prepare the object file to receive the table blocks.

cdu$prepare_object_file(.cld_fab);

! Prepare the listing file, if any, to receive the listing.

cdu$prepare_listing_file(.cld_fab);

! Parse the CLD file into an intermediate representation.

cdu$cld();

! If no syntax errors were discovered, then generate all of the CLI
! table blocks from the intermediate representation.

if .cdu$gl_cld_errors eqlu 0 then
        cdu$generate_table_blocks();

! If no errors of any kind were discovered, then write the object file.

if .cdu$gl_cld_errors eqlu 0 then
        cdu$write_object_file();

! Finish up the listing file.
```

```
;  281          0687  2 cdu$report_listing_trailer();
;  282          0688  2
;  283          0689  2 return;
;  284          0690  2
;  285          0691  1 END;



                                          000C 00000        .ENTRY  CDU$OBJECT_MODE, Save R2,R3                 ; 0643
                       53 00000000G   00    9E 00002        MOVAB   CDU$GL_CLD_ERRORS, R3
                       50             01    90 00009        MOVB    #1, FIRST_CLD                               ; 0644
             00000000G 00             00    FB 0000C        CALLS   #0, CDU$PREPARE_NEW_TABLE                   ; 0654
             00000000G 00             00    FB 00013        CALLS   #0, CDU$OPEN_NEXT_CLD                       ; 0658
                       52             50    D0 0001A        MOVL    R0, CLD_FAB
                                      36    13 0001D        BEQL    3$                                          ; 0659
                                      52    DD 0001F        PUSHL   CLD_FAB                                     ; 0664
             00000000G 00             01    FB 00021        CALLS   #1, CDU$PREPARE_OBJECT_FILE
                                      52    DD 00028        PUSHL   CLD_FAB                                     ; 0668
             00000000G 00             01    FB 0002A        CALLS   #1, CDU$PREPARE_LISTING_FILE
             00000000G 00             00    FB 00031        CALLS   #0, CDU$CLD                                 ; 0672
                                      63    D5 00038        TSTL    CDU$GL_CLD_ERRORS                           ; 0677
                                      07    12 0003A        BNEQ    1$
             00000000G 00             00    FB 0003C        CALLS   #0, CDU$GENERATE_TABLE_BLOCKS               ; 0678
                                      63    D5 00043 1$:    TSTL    CDU$GL_CLD_ERRORS                           ; 0682
                                      07    12 00045        BNEQ    2$
             00000000G 00             00    FB 00047        CALLS   #0, CDU$WRITE_OBJECT_FILE                   ; 0683
             00000000G 00             00    FB 0004E 2$:    CALLS   #0, CDU$REPORT_LISTING_TRAILER              ; 0687
                                         04 00055 3$:       RET                                                 ; 0691

; Routine Size:  86 bytes,    Routine Base:  $CODE$ + 00E6
```

```
1 !++
1 ! Description:  This routine handles /REPLACE mode, which is the fundamental
1 !               mode by which a user adds or replaces command definitions.
1 !               We compile a set of CLD files and add/replace the
1 !               definitions to an existing CLI table specified by the user.
1 !               When compilation is complete, we create a new CLI table
1 !               with all the resulting definitions.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$replace_mode         : novalue
2 = BEGIN
2
2 local
2         cld_fab: pointer,
2         errors: boolean initial(false);
2
2
2 ! Call a routine to prepare the input CLI table for modification.
2
2 cdu$prepare_input_table();
2
2 ! Sit in a loop to compile each CLD file.  Open each file in turn, quiting
2 ! when we run out of files.
2
3 while (cld_fab = cdu$open_next_cld()) neqa 0 do (
3
3         ! Prepare the listing file, if any, to receive the listing.
3
3         cdu$prepare_listing_file(.cld_fab);
3
3         ! Parse the CLD file into its intermediate representation.
3
3         cdu$cld();
3
3         ! If no syntax errors were discovered, then generate all of the CLI
3         ! table blocks from the intermediate representation.
3
3         if .cdu$gl_cld_errors eqlu 0 then
3                 cdu$generate_table_blocks();
3
3         ! Remember if any errors occurred, so we won't write the new table.
3
3         if .cdu$gl_cld_errors nequ 0 then
3                 errors = true;
3
3         ! Clear away the intermediate representation to prepare for the
3         ! next CLD file.
3
3         cdu$free_all_nodes();
3
3         ! Finish up the listing file.
;  344            0749  3
;  345            0750  3               cdu$report_listing_trailer();
;  346            0751  2  );
;  347            0752  2
;  348            0753  2  ! If no errors were discovered, then write out the new CLI table.
;  349            0754  2
;  350            0755  2  if not .errors then
;  351            0756  2         cdu$write_output_table();
;  352            0757  2
;  353            0758  2  return;
;  354            0759  2
;  355            0760  1  END;



                                              001C 00000           .ENTRY  CDU$REPLACE_MODE, Save R2,R3,R4                ; 0707
                       54 00000000G   00  9E 00002           MOVAB   CDU$GL_CLD_ERRORS, R4                             ;
                                      53  94 00009           CLRB    ERRORS                                            ; 0708
              00000000G  00           00  FB 0000B           CALLS   #0, CDU$PREPARE_INPUT_TABLE                       ; 0717
              00000000G  00           00  FB 00012 1$:       CALLS   #0, CDU$OPEN_NEXT_CLD                             ; 0722
                         52           50  D0 00019           MOVL    R0, CLD_FAB                                       ;
                                      32  13 0001C           BEQL    4$                                                ;
                                      52  DD 0001E           PUSHL   CLD_FAB                                           ; 0726
              00000000G  00           01  FB 00020           CALLS   #1, CDU$PREPARE_LISTING_FILE                      ;
              00000000G  00           00  FB 00027           CALLS   #0, CDU$CLD                                       ; 0730
                                      64  D5 0002E           TSTL    CDU$GL_CLD_ERRORS                                 ; 0735
                                      07  12 00030           BNEQ    2$                                                ;
              00000000G  00           00  FB 00032           CALLS   #0, CDU$GENERATE_TABLE_BLOCKS                     ; 0736
                                      64  D5 00039 2$:       TSTL    CDU$GL_CLD_ERRORS                                 ; 0740
                                      03  13 0003B           BEQL    3$                                                ;
                         53           01  90 0003D           MOVB    #1, ERRORS                                        ; 0741
              00000000G  00           00  FB 00040 3$:       CALLS   #0, CDU$FREE_ALL_NODES                            ; 0746
              00000000G  00           00  FB 00047           CALLS   #0, CDU$REPORT_LISTING_TRAILER                    ; 0750
                                      C2  11 0004E           BRB     1$                                                ; 0722
                         07           53  E8 00050 4$:       BLBS    ERRORS, 5$                                        ; 0755
              00000000G  00           00  FB 00053           CALLS   #0, CDU$WRITE_OUTPUT_TABLE                        ; 0756
                                          04 0005A 5$:       RET                                                       ; 0760

; Routine Size:  91 bytes,    Routine Base:  $CODE$ + 013C
```

```
1 !++
1 ! Description:  This routine handles /SYMBOLS mode, in which the user wants to
1 !               generate a symbol table file from a set of CLD files.  The
1 !               symbol table file is needed when commands make use of the
1 !               old CLI interface.  The symbols define the qualifier and
1 !               keyword numbers for use with the old CLI callbacks.
1 !
1 !               In this mode, no CLI table blocks are generated.
1 !
1 ! Parameters:   None.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:
1 !--
1
1 GLOBAL ROUTINE cdu$symbols_mode          : novalue
2 = BEGIN
2
2 local
2         symbols_written: boolean initial(false);
2
2
2 ! Sit in a loop to compile each CLD file.  Open each file in turn, quiting
2 ! when we run out of files.
2
3 while cdu$open_next_cld() neqa 0 do (
3
3         ! Parse the CLD file into an intermediate representation.
3
3         cdu$cld();
3
3         ! If no syntax errors were discovered, then add the symbols from
3         ! this CLD to the symbol table file.
3
4         if .cdu$gl_cld_errors eqlu 0 then (
4                 cdu$write_symbol_table_file();
4                 symbols_written = true;
3         );
3
3         ! Clear away the intermediate representation to prepare for the
3         ! next CLD file.
3
3         cdu$free_all_nodes();
2 );
2
2 ! Close out the symbol table file if we ever wrote any.
2
2 if .symbols_written then
2         cdu$close_symbol_table_file();
2
2 return;
2
1 END;
```

```
                                             0004 00000          .ENTRY  CDU$SYMBOLS_MODE, Save R2                    ; 0777
                                           52  94 00002          CLRB    SYMBOLS_WRITTEN                              ; 0778
                 00000000G  00             00  FB 00004 1$:      CALLS   #0, CDU$OPEN_NEXT_CLD                        ; 0787
                                           50  D5 0000B          TSTL    R0                                           ;
                                           22  13 0000D          BEQL    3$                                           ;
                 00000000G  00             00  FB 0000F          CALLS   #0, CDU$CLD                                  ; 0791
                                00000000G  00  D5 00016          TSTL    CDU$GL_CLD_ERRORS                            ; 0796
                                           0A  12 0001C          BNEQ    2$                                           ;
                 00000000G  00             00  FB 0001E          CALLS   #0, CDU$WRITE_SYMBOL_TABLE_FILE              ; 0797
                            52             01  90 00025          MOVB    #1, SYMBOLS_WRITTEN                          ; 0798
                 00000000G  00             00  FB 00028 2$:      CALLS   #0, CDU$FREE_ALL_NODES                       ; 0804
                                           D3  11 0002F          BRB     1$                                           ; 0787
                            07             52  E9 00031 3$:      BLBC    SYMBOLS_WRITTEN, 4$                          ; 0809
                 00000000G  00             00  FB 00034          CALLS   #0, CDU$CLOSE_SYMBOL_TABLE_FILE              ; 0810
                                               04 0003B 4$:      RET                                                  ; 0814

; Routine Size:  60 bytes,    Routine Base:  $CODE$ + 0197
```

```
1 !++
1 ! Description:  This routine is called to report an error from an RMS
1 !               operation.
1 !
1 ! Parameters:   message          By value, a message status code used for the
1 !                                first line of the message.  It is assumed
1 !                                to take a single !AS $FAO argument, the file
1 !                                spec.
1 !               rms_block        By reference, a FAB or RAB which contains
1 !                                the error status code.
1 !
1 ! Returns:      Nothing.
1 !
1 ! Notes:        This routine assumes that all FABs have associated NAM
1 !               blocks.
1 !--
1
1 GLOBAL ROUTINE cdu$report_rms_error(message: long,
1                                     rms_block: pointer)         : novalue
2 = BEGIN
2
2 local
2         fab: pointer,
2         nam: pointer,
2         file_spec: descriptor;
2
2
2 ! Pick up a pointer to the FAB and NAM blocks.
2
2 fab = (if .rms_block[fab$b_bid] eqlu fab$c_bid then .rms_block else .rms_block[rab$l_fab]);
2 nam = .fab[fab$l_nam];
2
2 ! We need to find a file spec which can be included in the first message
2 ! line.  Use the one which is most complete.
2
2 if .nam[nam$b_rsl] nequ 0 then
3         build_descriptor(file_spec, .nam[nam$b_rsl],.nam[nam$l_rsa])
2 else if .nam[nam$b_esl] nequ 0 then
3         build_descriptor(file_spec, .nam[nam$b_esl],.nam[nam$l_esa])
2 else
2         build_descriptor(file_spec, .fab[fab$b_fns],.fab[fab$l_fna]);
2 str$trim(file_spec,file_spec,file_spec);
2
2 ! Signal the error stored in the RMS block.
2
2 if .rms_block[fab$b_bid] eqlu fab$c_bid then
2         signal(.message,1,file_spec, .rms_block[fab$l_sts],.rms_block[fab$l_stv])
2 else
2         signal(.message,1,file_spec, .rms_block[rab$l_sts],.rms_block[rab$l_stv]);
2
2 return;
2
1 END;
```

```
                                             000C 00000            .ENTRY  CDU$REPORT_RMS_ERROR, Save R2,R3              ; 0832
                        5E              08   C2 00002            SUBL2   #8, SP                                         ;
                        52        08    AC   D0 00005            MOVL    RMS_BLOCK, R2                                  ; 0844
                                        53   D4 00009            CLRL    R3                                             ;
                        03              62   91 0000B            CMPB    (R2), #3                                       ;
                                        07   12 0000E            BNEQ    1$                                             ;
                                        53   D6 00010            INCL    R3                                             ;
                        51              52   D0 00012            MOVL    R2, FAB                                        ;
                                        04   11 00015            BRB     2$                                             ;
                        51        3C    A2   D0 00017 1$:        MOVL    60(R2), FAB                                    ;
                        50        28    A1   D0 0001B 2$:        MOVL    40(FAB), NAM                                   ; 0845
                                  03    A0   95 0001F            TSTB    3(NAM)                                         ; 0850
                                        0E   13 00022            BEQL    3$                                             ;
                        6E        03    A0   9B 00024            MOVZBW  3(NAM), FILE_SPEC                              ; 0851
                                  02    AE   B4 00028            CLRW    FILE_SPEC+2                                    ;
                 04     AE        04    A0   D0 0002B            MOVL    4(NAM), FILE_SPEC+4                            ;
                                        1F   11 00030            BRB     5$                                             ; 0850
                                  0B    A0   95 00032 3$:        TSTB    11(NAM)                                        ; 0852
                                        0E   13 00035            BEQL    4$                                             ;
                        6E        0B    A0   9B 00037            MOVZBW  11(NAM), FILE_SPEC                             ; 0853
                                  02    AE   B4 0003B            CLRW    FILE_SPEC+2                                    ;
                 04     AE        0C    A0   D0 0003E            MOVL    12(NAM), FILE_SPEC+4                           ;
                                        0C   11 00043            BRB     5$                                             ; 0852
                        6E        34    A1   9B 00045 4$:        MOVZBW  52(FAB), FILE_SPEC                             ; 0855
                                  02    AE   B4 00049            CLRW    FILE_SPEC+2                                    ;
                 04     AE        2C    A1   D0 0004C            MOVL    44(FAB), FILE_SPEC+4                           ;
                                        5E   DD 00051 5$:        PUSHL   SP                                             ; 0856
                                  04    AE   9F 00053            PUSHAB  FILE_SPEC                                      ;
                                  08    AE   9F 00056            PUSHAB  FILE_SPEC                                      ;
          00000000G     00              03   FB 00059            CALLS   #3, STR$TRIM                                   ;
                        7E        08    A2   7D 00060            MOVQ    8(R2), -(SP)                                   ; 0863
                                  08    AE   9F 00064            PUSHAB  FILE_SPEC                                      ;
                                        01   DD 00067            PUSHL   #1                                             ;
                                  04    AC   DD 00069            PUSHL   MESSAGE                                        ;
          00000000G     00              05   FB 0006C            CALLS   #5, LIB$SIGNAL                                 ;
                                             04 00073            RET                                                    ; 0867

; Routine Size:  116 bytes,    Routine Base:  $CODE$ + 01D3


;  465          0868  1 END
;  466          0869  0 ELUDOM



                                                                   .EXTRN  LIB$SIGNAL

;                                PSECT SUMMARY
;
;       Name                  Bytes                         Attributes
;
;   $PLIT$                      116  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $OWN$                        12  NOVEC,  WRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   $CODE$                      583  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;                                  Library Statistics
;
;                                                    -------- Symbols --------       Pages        Processing
;       File                                         Total     Loaded   Percent      Mapped       Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1                     18619         19         0      1000           00:01.9



;                                        COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:MAIN/OBJ=OBJ$:MAIN MSRC$:MAIN/UPDATE=(ENH$:MAIN)

; Size:          583 code + 128 data bytes
; Run Time:         00:14.5
; Elapsed Time:     00:53.0
; Lines/CPU Min:     3608
; Lexemes/CPU-Min: 17373
; Memory Used:  116 pages
; Compilation Complete
```

0044 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

SYMBOLS
LIS

NODES
LIS

OBJECT
LIS

PARSE1
LIS

PARSE3
LIS

ROUTINES
LIS

LISTING
LIS

MAIN
LIS

TABLE
LIS

PARSE2
LIS